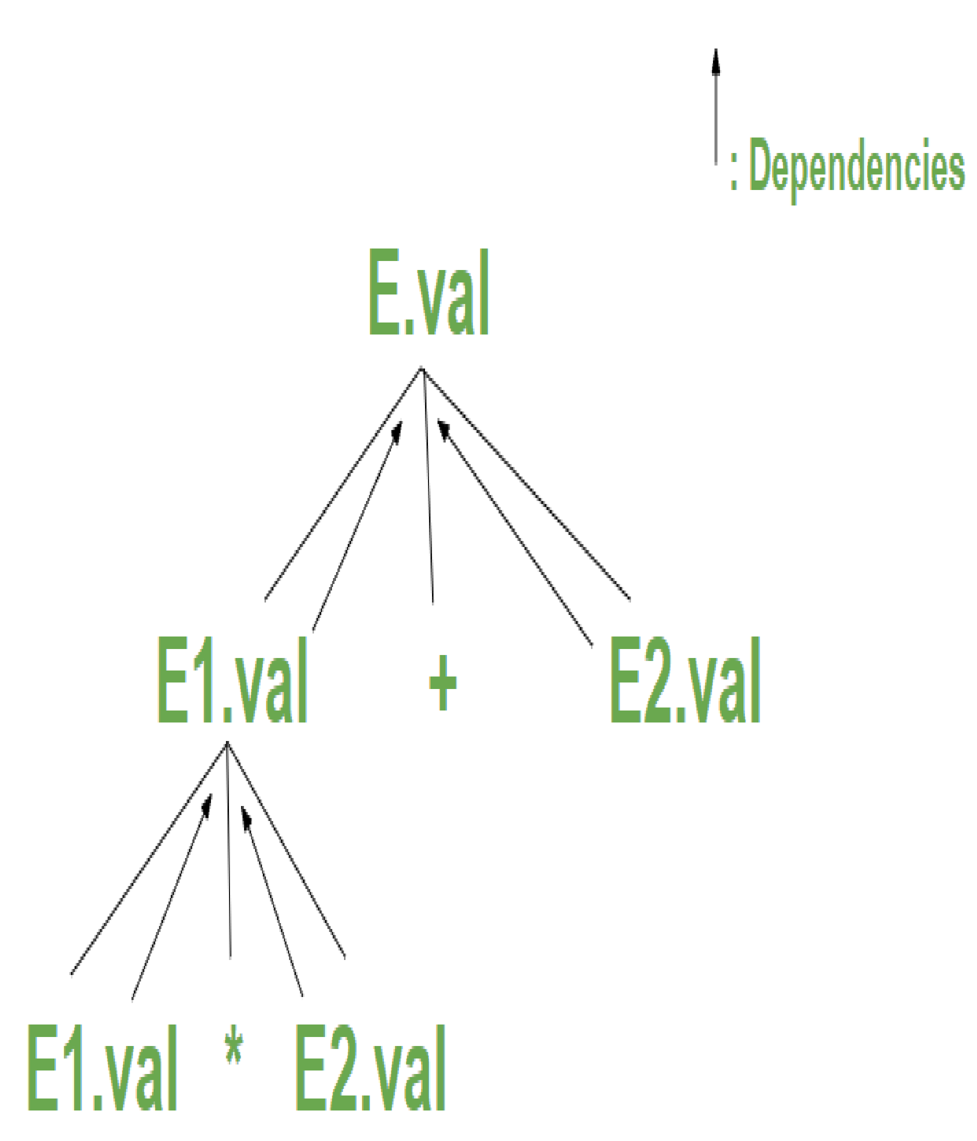
Course guided by Prof Mr. Sameer Iqbal Tamboli

A Brief Overview of Dependency Graphs

Presented by : Anushka Harshavadan Nevgi
TY BTECH CSE ‘A2’
Roll no: 31



What is a Dependency Graph?

A dependency graph represents the flow of information among attributes in a parse tree.

Purpose:

1. Determines evaluation order for attributes.
2. Prevents incorrect execution sequences affecting program meaning.
3. Helps identify components.

Structure:

1. Nodes (representing statements/variables).
2. Directed edges (showing dependencies).

Types of Dependencies in Compiler Design

Dependencies are classified into:

1. Data Dependency
2. Control Dependency
3. Flow Dependency
4. Anti-Dependency
5. Output Dependency
6. Control Dependency (Reiterated due to a separate definition)

Each type affects execution order and parallelization.

Data Dependency

1. Occurs when one instruction's output is used as input by another.
2. Represents the flow of data between instructions.
3. Must be preserved to maintain program correctness.
4. Also known as "True Dependency" or "Read After Write (RAW)" dependency.

Example of Data Dependency:

- A = 5; B = A + 3;
- B depends on A's value.
- If A isn't assigned first, B causes an error.

Control Dependency

1. Arises when an instruction's execution depends on a condition.
2. Common in conditional statements (if-else, loops).
3. Affects program flow and instruction scheduling.
4. Can impact parallel execution and CPU pipelining.

Example of Control Dependency:

- if (x > 5) { y = 10; }
- y executes only if x > 5.
- If condition fails, y is skipped.

Flow Dependency

1. Occurs when a statement reads data produced by a previous statement.
2. Ensures correct sequencing of dependent instructions.
3. Prevents reordering to maintain data integrity.
4. A subset of data dependency focused on execution flow.

Example of Flow Dependency:

- A = 5; B = A;
- B reads the value of A.
- Changing order affects correctness.

Anti Dependency

1. Occurs when a statement writes to a variable that was previously read.
2. Order-sensitive: Writing too early can cause incorrect results.
3. Affects parallel execution and instruction scheduling.
4. Common in pipeline hazards in processors.

Example of Anti Dependency:

- A = 5; B = A; A = 10;
- B depends on A's original value before modification.
- If A = 10 is executed first, B gets incorrect data.

Output Dependency

1. Occurs when two instructions write to the same variable.
2. Final value depends on execution order.
3. Can lead to unpredictable results in parallel execution.
4. Needs careful scheduling to avoid overwriting values.
5. It is also called as Write After Write

Example of Output Dependency:

- A = 5; A = 10;
- Final value of A depends on execution order.
- If reversed, result would be different.

Control Dependency (Conditional Execution)

1. Arises when an instruction is executed only if a condition is met.
2. Found in branching and decision-making constructs.
3. Impacts CPU performance due to branch prediction.
4. Must be handled properly to avoid execution stalls.

Example of Control Dependency (Conditional Execution):

- if (x > 5) { y = 10; } z = 20;
- y executes only if x > 5, while z executes regardless.

Advantages	Disadvantages
Ensures Correct Execution – Maintains proper instruction order (Data Dependency).	Slows Execution – Instructions wait for dependent data.
Optimizes Performance – Helps in efficient code execution (Flow Dependency).	Limits Parallelism – Conditional checks restrict execution.
Controls Execution Flow – Ensures correct conditional execution (Control Dependency).	Causes Pipeline Stalls – Dependencies lead to delays.
Prevents Conflicts – Avoids overwriting before reading (Anti-Dependency).	Increases Complexity – Optimization becomes harder.
Maintains Output Consistency – Ensures the correct final result (Output Dependency).	Hard to Modify – Changes may require reordering.

thank
you